System Design: Go Monolithic REST API Backend

1. **Executive Summary**

This document outlines the architecture for migrating the current Supabase-based backend to a Go monolithic REST API. The platform serves system architects and backend developers for creating architectural diagrams, collaborative whiteboards, and mind maps.

2. Functional Requirements

2.1 Authentication & Authorization

Feature	Description
Email/Password Auth	User registration, login, logout with JWT tokens
Session Management	Token refresh, revocation, multi-device support
Role-Based Access	Owner, Admin, Member, Viewer roles per workspace
Password Reset	Email-based password recovery flow

2.2 Workspace Management

Feature	Description
CRUD Workspaces	Create, read, update, delete workspaces
Member Management	Invite, remove, update roles for members
Email Invitations	Send invite emails with secure tokens (7-day expiry)
Workspace Settings	Color, tags, description customization

2.3 Diagram & Whiteboard Management

Feature	Description
CRUD Diagrams	Create, read, update, delete diagrams/whiteboards
Diagram Types	Flowchart, Sequence, Class, ER, State, Gantt, C4, Mind Map
Public/Private	Toggle diagram visibility
Workspace Scoping	Associate diagrams with workspaces
Image Storage	Store diagram thumbnails/exports

2.4 AI Diagram Generation

Feature	Description
Text-to-Diagram	Convert natural language to Mermaid syntax
Multi-Model Support	Gemini, GPT integration via gateway

2.5 Real-Time Collaboration

Feature	Description
Cursor Tracking	Live cursor positions for collaborators
Canvas Sync	Real-time canvas state broadcasting
Presence	Active user indicators
Session Management	Join/leave collaboration sessions

2.6 Comments

Feature	Description
CRUD Comments	Add, edit, delete comments on diagrams
Real-Time Updates	Live comment feed

3. Non-Functional Requirements

Category	Requirement	Target
Performance	API response time	< 100ms (p95)
Scalability	Concurrent users	10,000+
Availability	Uptime	99.9%
Security	Data encryption	TLS 1.3, AES-256 at rest
Consistency	Data integrity	ACID transactions
Latency	WebSocket latency	< 50ms
Storage	File upload limit	10MB per file
Rate Limiting	API requests	100 req/min per user

4. Database Design

4.1 Schema (PostgreSQL)

```
-- USERS
CREATE TABLE users (
   id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
   email VARCHAR(255) UNIQUE NOT NULL,
   password_hash VARCHAR(255) NOT NULL,
   email_verified BOOLEAN DEFAULT FALSE,
   created_at TIMESTAMPTZ DEFAULT NOW(),
   updated_at TIMESTAMPTZ DEFAULT NOW()
);

-- WORKSPACES
CREATE TABLE workspaces (
   id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
   name VARCHAR(255) NOT NULL,
   description TEXT,
   color VARCHAR(20),
   tags TEXT[],
   created_by UUID REFERENCES users(id) ON DELETE SET NULL,
   created_at TIMESTAMPTZ DEFAULT NOW(),
   updated_at TIMESTAMPTZ DEFAULT NOW()
);

-- WORKSPACE_MEMBERS
CREATE TABLE workspace_members (
   id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
   workspace_id UUID REFERENCES workspaces(id) ON DELETE CASCADE,
   user_id UUID REFERENCES users(id) ON DELETE CASCADE,
   role VARCHAR(20) CHECK (role IN ('owner', 'admin', 'member', 'viewer')),
   invited_by UUID REFERENCES users(id),
   joined_at TIMESTAMPTZ DEFAULT NOW(),
   UNIQUE(workspace_id, user_id)
);

-- WORKSPACE_INVITATIONS
CREATE TABLE workspace_invitations (
   id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
   workspace_id UUID REFERENCES workspaces(id) ON DELETE CASCADE,
   email VARCHAR(255) NOT NULL,
   role VARCHAR(20) DEFAULT 'member',
   token UUID UNIQUE DEFAULT gen_random_uuid(),
   invited_by UUID REFERENCES users(id),
   expires_at TIMESTAMPTZ DEFAULT NOW() + INTERVAL '7 days',
   created_at TIMESTAMPTZ DEFAULT NOW(),
   UNIQUE(workspace_id, email)
);

-- DIAGRAMS
CREATE TABLE diagrams (
   id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
   title VARCHAR(255) NOT NULL,
   content TEXT NOT NULL,
   diagram_type VARCHAR(50) NOT NULL,
   image_url TEXT,
   is_public BOOLEAN DEFAULT FALSE,
   user_id UUID REFERENCES users(id) ON DELETE SET NULL,
   workspace_id UUID REFERENCES workspaces(id) ON DELETE SET NULL,
   created_at TIMESTAMPTZ DEFAULT NOW(),
   updated_at TIMESTAMPTZ DEFAULT NOW()
);

-- COMMENTS
CREATE TABLE comments (
   id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
   diagram_id UUID REFERENCES diagrams(id) ON DELETE CASCADE,
   user_id UUID REFERENCES users(id) ON DELETE CASCADE,
   comment_text TEXT NOT NULL,
   created_at TIMESTAMPTZ DEFAULT NOW(),
   updated_at TIMESTAMPTZ DEFAULT NOW()
);

-- COLLABORATION_SESSIONS (ephemeral, consider Redis)
CREATE TABLE collaboration_sessions (
   id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
   diagram_id UUID REFERENCES diagrams(id) ON DELETE CASCADE,
   user_id UUID REFERENCES users(id) ON DELETE CASCADE,
   cursor_position JSONB,
   last_seen TIMESTAMPTZ DEFAULT NOW()
);

-- INDEXES
CREATE INDEX idx_diagrams_user ON diagrams(user_id);
CREATE INDEX idx_diagrams_workspace ON diagrams(workspace_id);
CREATE INDEX idx_workspace_members_user ON workspace_members(user_id);
CREATE INDEX idx_comments_diagram ON comments(diagram_id);
CREATE INDEX idx_invitations_token ON workspace_invitations(token);
CREATE INDEX idx_invitations_email ON workspace_invitations(email);
```

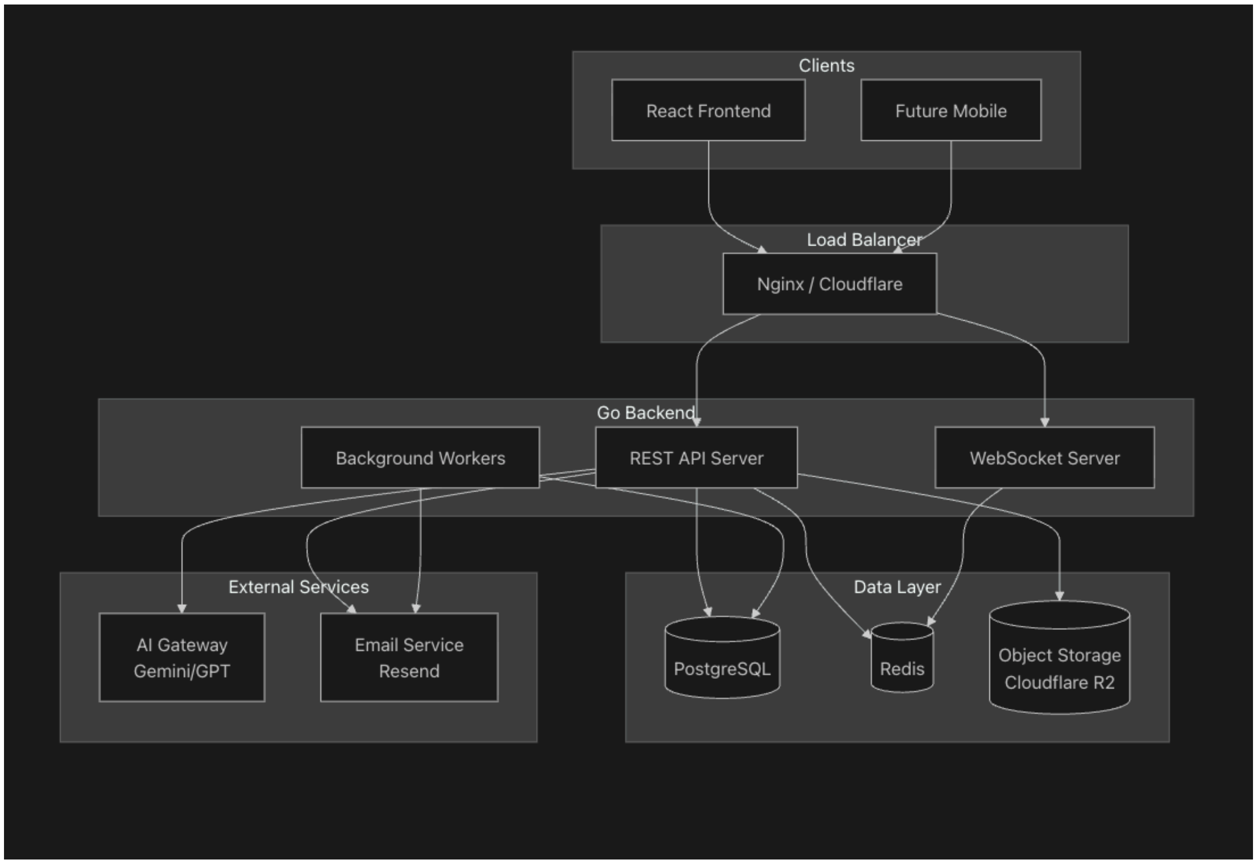
4.2 Database Layer Recommendations

Component	Technology	Rationale
Primary DB	PostgreSQL 16	ACID, JSON support, proven scale
Connection Pool	pgxpool	High-performance Go driver
Migrations	golang-migrate	Version-controlled schema changes

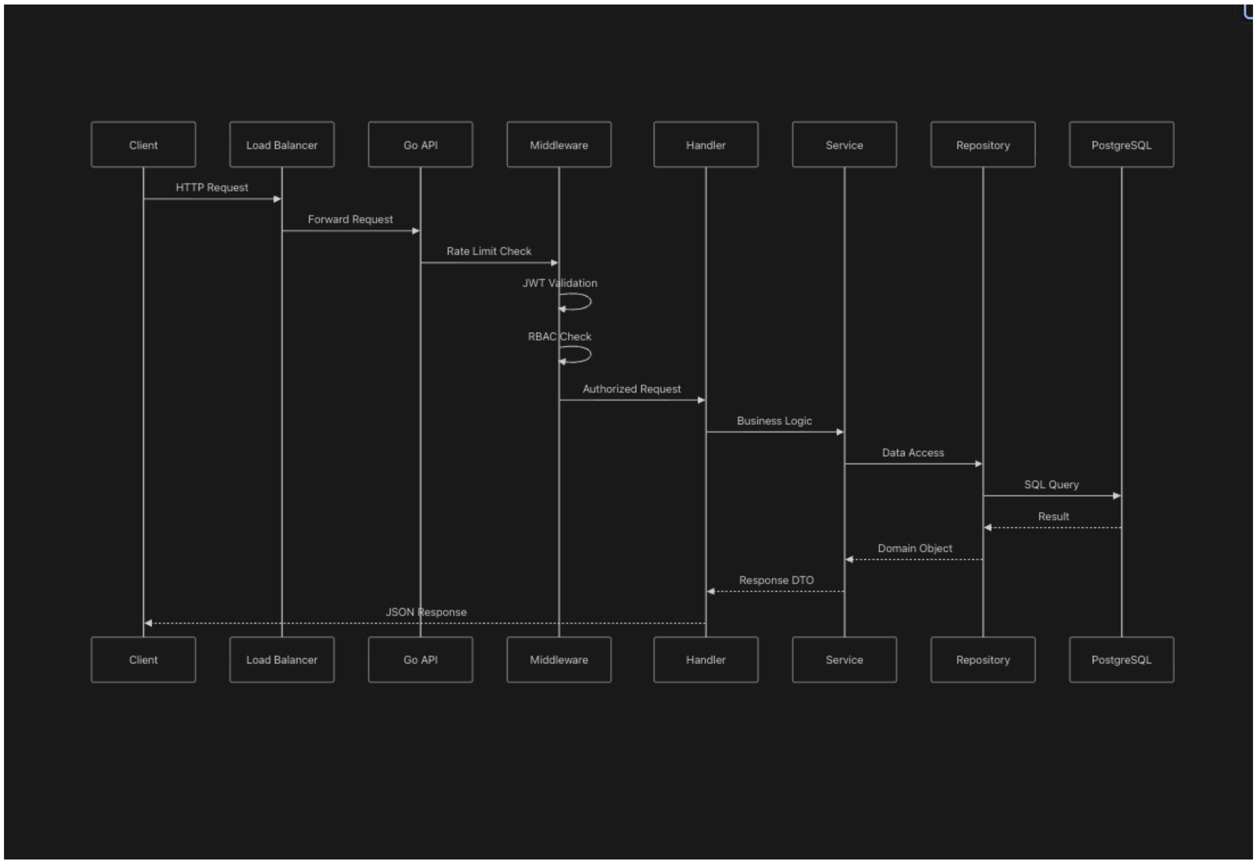
Query Builder	sqlc or GORM	Type-safe queries
Caching	Redis	Session data, collaboration state

5. Architecture

5.1 High-Level Architecture



5.2 API Architecture Flow



6. API Endpoints

6.1 Authentication

Method	Endpoint	Description
POST	/api/v1/auth/register	User registration
POST	/api/v1/auth/login	User login
POST	/api/v1/auth/logout	User logout
POST	/api/v1/auth/refresh	Refresh JWT

POST	/api/v1/auth/forgot-password	Request password reset
POST	/api/v1/auth/reset-password	Reset password

6.2 Workspaces

Method	Endpoint	Description
GET	/api/v1/workspaces	List user's workspaces
POST	/api/v1/workspaces	Create workspace
GET	/api/v1/workspaces/:id	Get workspace details
PUT	/api/v1/workspaces/:id	Update workspace
DELETE	/api/v1/workspaces/:id	Delete workspace
GET	/api/v1/workspaces/:id/members	List members
POST	/api/v1/workspaces/:id/invitations	Invite member
DELETE	/api/v1/workspaces/:id/members/:userId	Remove member
PUT	/api/v1/workspaces/:id/members/:userId	Update member role
POST	/api/v1/invitations/accept	Accept invitation

6.3 Diagrams

Method	Endpoint	Description
GET	/api/v1/diagrams	List diagrams (with filters)
POST	/api/v1/diagrams	Create diagram

GET	/api/v1/diagrams/:id	Get diagram
PUT	/api/v1/diagrams/:id	Update diagram
DELETE	/api/v1/diagrams/:id	Delete diagram
POST	/api/v1/diagrams/:id/image	Upload diagram image
GET	/api/v1/diagrams/:id/comments	Get comments
POST	/api/v1/diagrams/:id/comments	Add comment

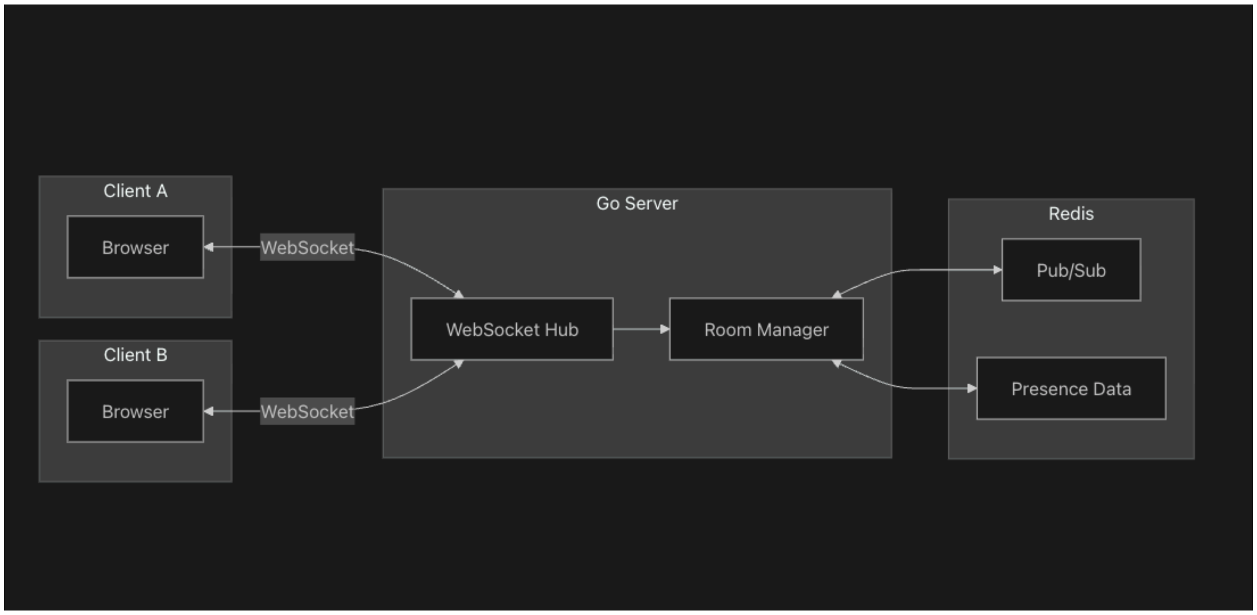
6.4 AI Generation

Method	Endpoint	Description
POST	/api/v1/ai/generate-diagram	Generate diagram from text

6.5 Real-Time (WebSocket)

Endpoint	Description
/ws/collaboration/:diagramId	Real-time collaboration

7. Real-Time Collaboration Architecture



Message Types:

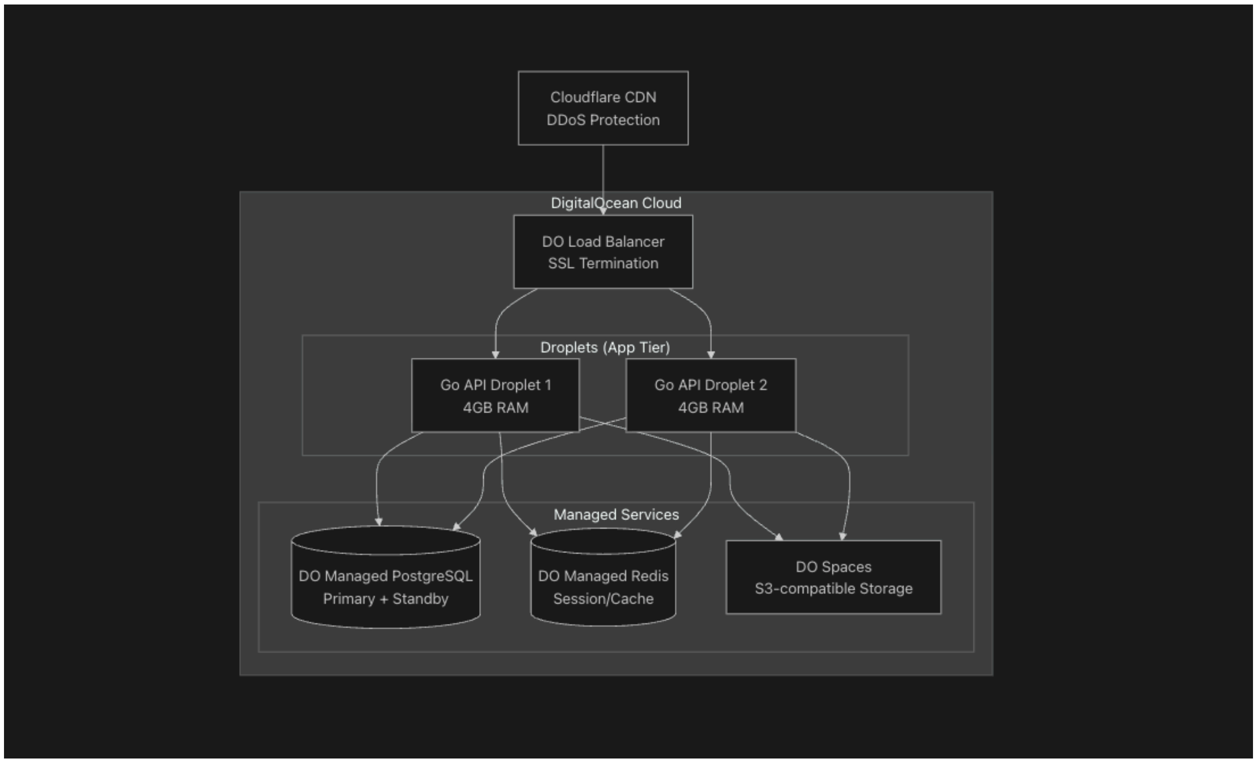
- cursor_update - Cursor position changes
- canvas_update - Canvas state changes
- user_joined - New collaborator
- user_left - Collaborator disconnected

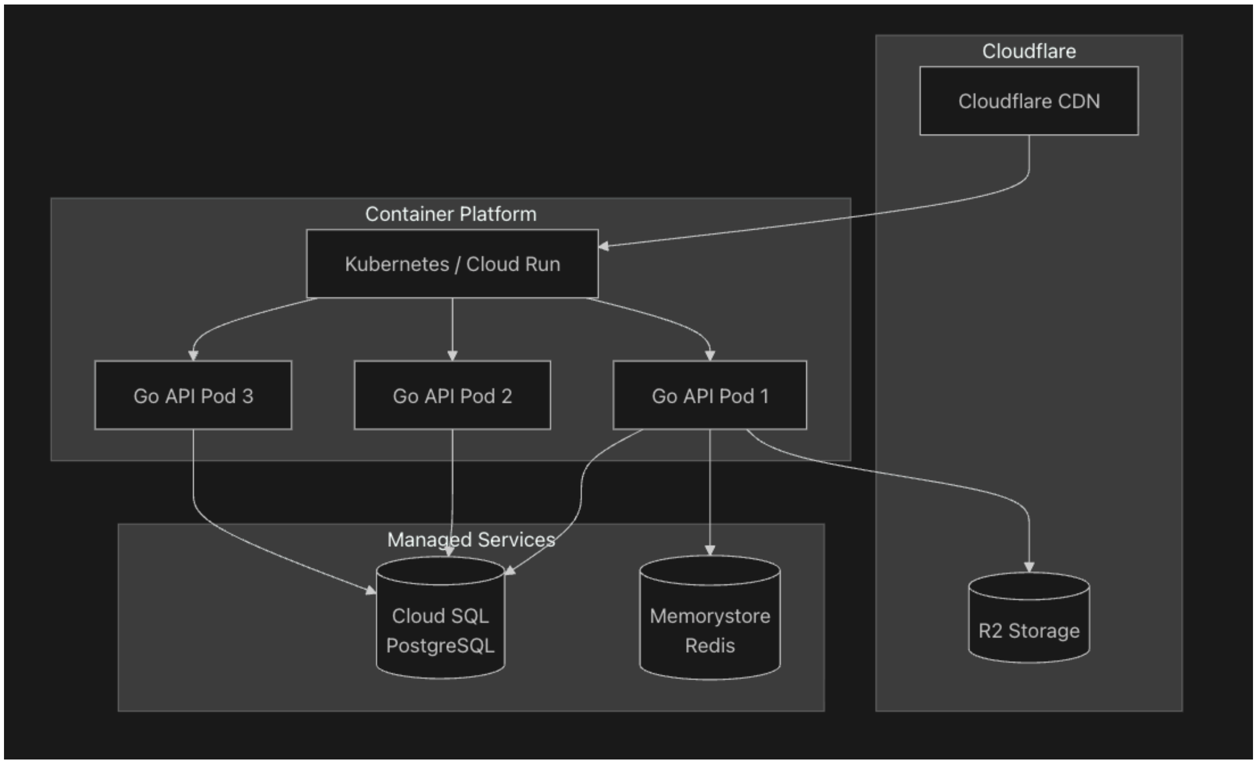
---

8. Security Considerations

Layer	Implementation
Transport	TLS 1.3 everywhere
Authentication	JWT with RS256, short-lived access tokens (15min), refresh tokens (7 days)
Authorization	Middleware-based RBAC checking workspace membership/roles
Input Validation	Go validator, sanitized SQL via parameterized queries
Rate Limiting	Redis-backed token bucket (100 req/min)
CORS	Whitelist allowed origins
File Upload	Type validation, size limits, virus scanning
Secrets	Environment variables, Vault for production

9. Deployment Architecture


Cloudflare CDN
DDoS Protection
DigitalOcean Cloud
DO Load Balancer
SSL Termination
Droplets (App Tier)
Go API Droplet 1
4GB RAM
Go API Droplet 2
4GB RAM
Managed Services
DO Managed PostgreSQL
Primary + Standby
DO Managed Redis
Session/Cache
DO Spaces
S3-compatible Storage


Cloudflare
Cloudflare CDN
Container Platform
Kubernetes / Cloud Run
Go API Pod 3
Go API Pod 2
Go API Pod 1
Managed Services
Cloud SQL
PostgreSQL
Memorystore
Redis
R2 Storage

10. Migration Strategy

Phase 1: Parallel Development (2-3 weeks)

- Set up Go project structure
- Implement core auth and CRUD endpoints
- Dual-write to both backends

Phase 2: Feature Parity (3-4 weeks)

- Migrate all edge functions logic
- Implement WebSocket collaboration
- Set up file storage with R2

Phase 3: Testing & Cutover (2 weeks)

- Load testing
- Feature flag rollout (10% → 50% → 100%)
- DNS cutover

Phase 4: Cleanup (1 week)

- Remove Supabase dependencies from frontend
- Update frontend API client to new endpoints

---

11. Recommended Go Libraries

Purpose	Library
HTTP Router	chi or fiber
WebSocket	gorilla/websocket
Database	pgx + sqlc
Validation	go-playground/validator
JWT	golang-jwt/jwt
Config	viper
Logging	zerolog or zap

Migrations	golang-migrate
Testing	testify + httptest

Infrastructure Components

Component	DigitalOcean Service	Starting Spec	Monthly Cost
Load Balancer	DO Load Balancer	Small	~$12
API Servers	Droplets	2x 4GB/2vCPU	~$48
Database	Managed PostgreSQL	1GB RAM (dev) → 4GB (prod)	$15-60
Cache/Sessions	Managed Redis	1GB	~$15
File Storage	Spaces + CDN	250GB	~$5
Total (Start)			~$95/mo

Deployment Strategy

```
# Simple deployment with systemd + rsync or DO App Platform
deployment:
  method: "systemd service on Droplets"
  ci_cd: "GitHub Actions"

  steps:
    1. Build Go binary (linux/amd64)
    2. SCP binary to droplets
    3. Restart systemd service
    4. Health check
    5. Rollback on failure
```

Simpler Alternative: Use DigitalOcean App Platform which handles:

- Auto-scaling
- Zero-downtime deploys

- Built-in CI/CD from GitHub
- ~$12/mo for basic dyno

Completeness Checklist

Category	Covered?	Notes
Authentication	✅	JWT + refresh tokens, password hashing
Authorization	✅	RBAC (owner/admin/member/viewer)
Database Schema	✅	All 7 tables mapped
API Endpoints	✅	Full CRUD for all entities
Real-time Collab	✅	WebSocket with Redis Pub/Sub
AI Integration	✅	Diagram generation via external APIs
Email Service	⚠️	Need to add (Resend/SendGrid)
File Storage	✅	DO Spaces for exports/thumbnails
Rate Limiting	✅	Redis-based limiter
Logging/Monitoring	⚠️	Need to add observability stack
CI/CD Pipeline	⚠️	Need GitHub Actions workflow
Database Migrations	⚠️	Need migration tool (golang-migrate)

Missing Pieces to Add

1. Email Service - For workspace invitations (currently using Resend edge function)
2. Observability - Logging (Loki/Papertrail), Metrics (Prometheus), Tracing (optional)
3. CI/CD Pipeline - GitHub Actions workflow for build/test/deploy
4. Migration Tool - golang-migrate for schema versioning
5. API Documentation - OpenAPI spec for frontend integration
6. Health Checks - /health and /ready endpoints
7. Graceful Shutdown - Handle SIGTERM properly

Future Microservices Path

When you're ready to scale specific features:

```
Monolith → Extract high-load services first:
├── Real-time Service (WebSocket heavy)
├── AI Generation Service (GPU/async)
└── Keep REST API as monolith for CRUD
```